

Quicker Q-Learning in Multi-Agent Systems

Adrian K. Agogino

UCSC, NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035
adrian@email.arc.nasa.gov

Kagan Tumer

NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035
kagan@email.arc.nasa.gov

Abstract

Multi-agent learning in Markov Decisions Problems is challenging because of the presence of two credit assignment problems: 1) How to credit an action taken at time step t for rewards received at $t' > t$; and 2) How to credit an action taken by agent i considering the system reward is a function of the actions of all the agents. The first credit assignment problem is typically addressed with temporal difference methods such as Q-learning or TD(λ). The second credit assignment problem is typically addressed either by hand-crafting reward functions that assign proper credit to an agent, or by making certain independence assumptions about an agent's state-space and reward function. To address both credit assignment problems simultaneously, we propose the "Q Updates with Immediate Counterfactual Rewards-learning" (QUICR-learning) designed to improve both the convergence properties and performance of Q-learning in large multi-agent problems. Instead of assuming that an agent's value function can be made independent of other agents, this method suppresses the impact of other agents using counterfactual rewards. Results on multi-agent grid-world problems over multiple topologies show that QUICR-learning can achieve up to thirty fold improvements in performance over both conventional and local Q-learning in the largest tested systems.

1 Introduction

A critical issue in the multi-agent reinforcement learning process is the structural credit assignment problem: how to reward a single agent's action choices when the reward we intend to maximize is a function of all of the agents' actions. As an example consider how to reward construction rovers building a dome. This reward can be computed by measuring the performance of the rovers' actions over a series of episodes. For example in a single rover scenario, suppose the rover took action sequence, a_1 , for 100 episodes in simulations and the dome collapsed 50 times. Then the agent took action sequence, a_2 , for 100 episodes and the dome collapsed

30 times. Based on this evidence, we can say with high confidence that action sequence a_2 is better than action sequence a_1 . Now, consider the situation where a large team of rovers were used to build the dome. How does one evaluate the actions of a single rover in such a case? Suppose rover i took action sequence $a_{i,1}$ for 100 episodes and the dome collapsed 50 times. In addition rover i took action sequence, $a_{i,2}$, for 100 different episodes and the dome collapsed 30 times. Can we claim that action sequence $a_{i,2}$ was better than action sequence $a_{i,1}$? Not with the same confidence with which we claimed that action sequence a_2 was better than action sequence a_1 for the single rover case. Furthermore, our confidence will drop even further as the number of rovers in the system grows. This is because in a task with n homogeneous agents, on average, the choice of action sequence by agent i ($a_{i,1}$ or $a_{i,2}$) will likely have an impact of $\frac{1}{n}$ on the system performance. As a consequence, teasing out the impact $a_{i,1}$ will require far more iterations, as one needs to ascertain that $a_{i,1}$ was indeed a poorer choice than $a_{i,2}$, and not merely taken during episodes where the collective actions of the other agents were poor.

The difficulty here arises from evaluating an individual agent's actions using a function that, a-priori, all the agents impact equally. In such a case, a standard Q-learner is not equipped to handle the "structural" credit assignment problem of how to apportion the system credit to each individual agent. As an alternative, we present "Q Updates with Immediate Counterfactual Rewards learning" (QUICR-learning), which uses agent-specific rewards that suppress the impact of other agents. Rewards in QUICR-learning are both heavily agent-sensitive, making the learning task easier and aligned with the system level goal, ensuring that agents receiving high rewards are helping the system as a whole. These agent-specific reward functions are then used with standard temporal difference methods to create a learning method that is significantly faster than standard Q-learning in large multi-agent systems.

Currently the best multi-agent learning algorithms address the structural credit assignment problem by leveraging domain knowledge. In the robotic soccer for example, player specific subtasks are used, followed by tiling to provide good convergence properties [Stone *et al.*, 2005]. In a robot coordination problem for the foraging domain, specific rules inducing good division of labor are created [Jones and Mataric, 2003]. In domains where groups of agents can be assumed to

be independent, the task can be decomposed by learning a set basis functions used to represent the value function, where each basis only processed a small number of the state variables [Guestin *et al.*, 2002]. Also multi-agent Partially Observable Markov Decision Processes (POMDPs) can be simplified through piecewise linear rewards [Nair *et al.*, 2003]. There have also been several approaches to optimizing Q-learning in multi-agent systems that do not use independence assumptions. For a small number of agents, game theoretic techniques were shown to lead to a multi-agent Q-learning algorithm proven to converge [Hu and Wellman, 1998]. In addition, the equivalence between structural and temporal credit assignment was shown in [Agogino and Tumer, 2004], and methods based on Bayesian methods were shown to improve multi-agent learning by providing better exploration capabilities [Chalkiadakis and Boutilier, 2003]. In large systems game theory has addressed credit assignment in congestion problems with Vickrey Tolls [Vickrey, 1961].

In this paper, we present QUICR-learning which provides fast convergence in multi-agent learning domains, without assuming that the full system reward is linearly separable or requiring hand tuning based on domain knowledge. In Section 2 we discuss the temporal and structural credit assignment problems in multi-agent systems, and describe the QUICR-learning algorithm. In Section 3 we present results on two variants of a multi-agent gridworld problem, showing that QUICR-learning performs up to thirty times better than standard Q-learning in multi-agent problems.

2 Credit Assignment Problem

The multi-agent temporal credit assignment problem consists of determining how to assign rewards (e.g., credit) for a sequence of actions. Starting from current time step t , the undiscounted sum of rewards till a final time step T can be represented by:

$$R_t(s_t(a)) = \sum_{k=0}^{T-t} r_{t+k}(s_{t+k}(a)) \quad (1)$$

where a is a vector containing the actions of all agents at all time steps, $s_t(a)$ is the state function returning the state of all agents for a single time step, and $r_t(s)$ is the single-time-step reward function, which is a function of the states of all of the agents.

This reward is a function of all of the previous actions of all of the agents. Every reward is a function of the states of all the agents, and every state is a function of all the actions that preceded it (even though it is Markovian, the previous states ultimately depend on previous actions). In a system with n agents, on average $\frac{1}{2}n * T$ actions affect reward. Agents need to use this reward to evaluate their single action, yet even in the idealized domains presented Section 3, with one hundred agents and thirty-two time steps there are an average of 1600 actions affecting the reward!

2.1 Standard Q-Learning

Reinforcement learners such as Q-learning address how to assign credit of future rewards to an agent's current ac-

tion. The goal of Q-learning is to create a policy that maximizes the sum of future rewards, $R_t(s_t(a))$, from the current state [Kaelbling *et al.*, 1996; Sutton and Barto, 1998; Watkins and Dayan, 1992]. It does this by maintaining tables of Q-values, which estimate the expected sum of future rewards for a particular action in a particular state. In the TD(0) version of Q-learning, a Q-value, $Q(s_t, a_t)$, is updated with the following Q-learning rule¹:

$$\Delta Q(s_t, a_t) = \alpha(r_t + \max_a Q(s_{t+1}, a)) \quad (2)$$

The assumption with this update is that the action a_t is most responsible for the immediate reward r_t , but is somewhat less responsible for the sum of future rewards, $\sum_{k=1}^{T-t} r_{t+k}(s_{t+k}(a))$. This assumption is reasonable since rewards in the future are affected by uncertain future actions and noise in state transitions. Instead of using the sum of future rewards directly to update its table, Q-learning uses a Q-value from the next state entered as an estimate for those future rewards. Under benign assumptions, Q-values are shown to converge to the actual value of the future rewards [Watkins and Dayan, 1992].

Eventhough Q-learning addresses the temporal credit assignment problem (i.e., properly apportions the effects of all actions taken at other time steps to the current reward), the immediate reward in a multi-agent system still suffers from the structural credit assignment problem (i.e., the reward is still a function of all the agents' actions). Standard Q-learning does not address this structural credit assignment problem and an agent will get full credit for actions taken by all of the other agents. As a result when they are many agents, standard Q-learning is generally slow since it will take many episodes for an agent to figure out its impact on a reward it barely influences.

2.2 Local Q-Learning

One way to address the structural credit assignment problem and allow for fast learning is to assume that agents' actions are independent. Without this assumption, the immediate reward function for a multi-agent reward system may be a function of all the states:

$$r_t(s_{t,1}(a_1), s_{t,2}(a_1), \dots, s_{t,n}(a_n)) ,$$

where $s_{t,i}(a_i)$ is the state for agent i and is a function of only agent i 's previous actions. The number of states determining the reward grows linearly with the number of agents, while the number of actions that determine each state grows linearly with the number of time steps. To reduce the huge number of actions that affect this reward, often the reward is assumed to be linearly separable:

$$r_t(s_t) = \sum_i w_i r_{t,i}(s_{t,i}(a_i)) .$$

Then each agent receives a reward $r_{t,i}$ which is only a function of its action. Q-learning is then used to resolve the remaining temporal credit assignment problem. If the agents

¹This paper uses undiscounted learning to simplify notation, but all the algorithms also apply to discounted learning as well.

are actually independent, this method leads to a significant speedup in learning as an agent receives direct credit for its actions. If the agents are coupled, then the independence assumption still allows fast learning, but the agents will tend to converge to the wrong policy. With loose coupling the benefits of the assumption may still outweigh the costs when there are many agents. However, when agents are tightly coupled, the independence assumption may lead to unacceptable solutions and may even converge to a solution that is worse than random [Wolpert and Tumer, 2001].

2.3 QUICR-Learning

In this section we present QUICR-learning, a learning algorithm for multi-agent systems that does not assume that the system reward function is linearly separable. Instead it uses a mechanism for creating rewards that are a function of all of the agents, but still provide many of the benefits of hand-crafted rewards. Many hand-crafted multi-agent learning algorithms exploit detailed knowledge about a domain to provide agent rewards that allow the system to maximize a global reward. These rewards are designed to have two beneficial properties: they are “aligned” with the overall learning task and they have high “sensitivity” to the actions of the agent.

The first property of alignment means that when an agent maximizes its own reward it tends to maximize the overall system reward. Without this property, a large multi-agent system can lead to agents performing useless work, or worse, working at cross-purposes. Reward sensitivity means that an agent’s reward is more sensitive to its own actions than to other agents actions. This property is important for agents to learn quickly.

QUICR-learning is based on providing agents with rewards that are both aligned with the system goals and sensitive to the agent’s states. It aims to provide the benefits of hand-crafted algorithms without requiring detailed domain knowledge. In a task where the reward can be expressed as in Equation 1, let us introduce the difference reward (adapted from [Wolpert and Tumer, 2001]) given by:

$$D_t^i(s_t(a)) = R_t(s_t(a)) - R_t(s_t(a - a_{t,i}))$$

where $a - a_{t,i}$ denotes a counterfactual state where agent i has not taken the action it took in time step t (e.g., the action of agent i has been removed from the vector containing the actions of all the agents before the system state has been computed). Decomposing further, we obtain:

$$\begin{aligned} D_t^i(s_t(a)) &= \sum_{k=0}^{T-t} r_{t+k}(s_{t+k}(a)) - r_{t+k}(s_{t+k}(a - a_{t,i})) \\ &= \sum_{k=0}^{T-t} d_{t+k}(s_{t+k}(a), s_{t+k}(a - a_{t,i})). \end{aligned} \quad (3)$$

where $d_t(s_1, s_2) = r_t(s_1) - r_t(s_2)$. (We introduce the single time step “difference” reward d_t to keep the parallel between Equations 1 and 3). This reward is much more sensitive to an agent’s action than r_t since much of the effects of the other agents are subtracted out with the counterfactual [Wolpert and Tumer, 2001]. Unfortunately in general $d_t(s_1, s_2)$ is non-Markovian since the second parameter may depend of pre-

vious states, making its use troublesome in a learning task involving both a temporal and structural credit assignment.

In order to overcome this shortcoming of Equation 3, let us make the following two assumptions:

1. The counterfactual $a - a_{t,i}$ action moves agent i to an absorbing state, s_b that is independent of its current state.
2. The future state of agents other than agent i are not affected by the actions of agent i .

The first assumption forces us to compute a counterfactual state that is not necessarily a minor modification to agent i ’s current state. Therefore, differential function estimation techniques that rely on a small change in agent i ’s (e.g., Taylor series expansion) state cannot be used. However, each agent’s counterfactual state is for itself (e.g, not computed for other agents) and a single time step (e.g., the counterfactual states do not propagate through time). The second assumption holds in many multi-agent systems, since to reduce the state-space to manageable levels, agents often do not directly observe each other (though are still coupled through the reward).

Given these conditions, the counterfactual state for time $t+k$ is computed from the actual state at time $t+k$, by replacing the state of agent i at time t with s_b . Now the difference reward can be made into a Markovian function:

$$d_t^i(s_t) = r_t(s_t) - r_t(s_t - s_{t,i} + s_b), \quad (4)$$

where the expression $s_t - s_{t,i} + s_b$ denotes replacing agent i ’s state with state s_b .

Now the Q-learning rule can be applied to the difference reward, resulting in the QUICR-learning rule:

$$\begin{aligned} \Delta Q(s_t, a_t) &= \alpha(r_t(s_t) - r_t(s_t - s_{t,i} + s_b) \\ &\quad + \max_a Q(s_{t+1}, a)) \\ &= \alpha(d_t^i(s_t) + \max_a Q(s_{t+1}, a)) \end{aligned} \quad (5)$$

Note that since this learning rule is Q-learning, albeit applied to a different reward structure, it shares all the convergens properties of Q-learning. In order to show that Equation 5 leads to good system level behavior, we need to show that agent i maximizing $d_t^i(s_t)$ (e.g., following Equation 5) will maximize the system reward r_t . Note that by definition $(s_t - s_{t,i} + s_b)$ is independent of the actions of agent i , since it is formed by moving agent i to the absorbing state s_b from which it cannot emerge. This effectively means the partial differential of $d_t^i(s_t)$ with respect to agent i is²:

$$\begin{aligned} \frac{\partial}{\partial_i} d_t^i(s_t) &= \frac{\partial}{\partial_i} (r_t(s_t) - r_t(s_t - s_{t,i} + s_b)) \\ &= \frac{\partial}{\partial_i} r_t(s_t) - \frac{\partial}{\partial_i} r_t(s_t - s_{t,i} + s_b) \\ &= \frac{\partial}{\partial_i} r_t(s_t) - 0 \\ &= \frac{\partial}{\partial_i} r_t(s_t). \end{aligned} \quad (6)$$

²Though in this work we show this result for differentiable states, the principle applies to more general states, including discrete states.

Therefore any agent i using a learning algorithm to optimize $d_t^i(s_t)$ will also optimize $r_t(s_t)$. Furthermore, note that QUICR-learning converges not only to a globally desirable solution (e.g., it satisfies the first property of being aligned with the system level goal), but it also converges faster since the rewards are more sensitive to the actions of agent i because it removes much of the effects of the other agents through the counterfactual subtraction.

3 Multi-agent Grid World Experiments

We performed a series of simulations to test the performance of Q-Learning, Local Q-Learning and QUICR-Learning for multi-agent systems. We selected the multi-agent Grid World Problem, a variant of the standard Grid World Problem [Sutton and Barto, 1998]. In this problem, at each time step, the agent can move up, down, right or left one grid square, and receives a reward (possibly zero) after each move. The observable state space for the agent is its grid coordinate and the reward it receives depends on the grid square to which it moves. In the episodic version, which is the focus of this paper, the agent moves for a fixed number of time steps, and then is returned to its starting location.

In this paper we compare learning algorithms in a multi-agent version of the grid world problem. In this instance of the problem there are multiple agents navigating the grid simultaneously influencing each others' rewards. In this problem agents are rewarded for observing tokens located in the grid. Each token has a value between zero and one, and each grid square can have at most one token. When an agent moves into a grid square it observes a token and receives a reward for the value of the token. Rewards are only received on the first observation of the token. Future observations from the agent or other agents do not receive rewards in the same episode. If two agents move into the same square at the same time. More precisely, r_t is computed by summing the agents at the same location as unobserved tokens, weighted by the value of the tokens:

$$r_t(s_t) = \sum_i \sum_j V_j I_{s_t, i=L_j}^t \quad (7)$$

where I^t is the indicator function which returns one when an agent in state $s_{t,i}$ is in the location an unobserved token L_j . The global objective of the multi-agent Grid World Problem is to observe the highest aggregated value of tokens in a fixed number of time steps T .

3.1 Learning Algorithms

In each algorithm below, we use the TD(0) update rule. The standard Q-learning is based on the full reward r_t :

$$\Delta Q(s_t, a_t) = \alpha(r_t(s_t) + \max_a Q(s_{t+1}, a)) \quad (8)$$

Local Q-learning is only a function of the specific agent's own state:

$$\Delta Q_{loc}(s_t, a_t) = \alpha(\sum_j V_j I_{s_t, i=L_j}^{t,i} + \max_a Q_{loc}(s_{t+1}, a)).$$

QUICR-learning instead updates with a reward that is a function of all of the states, but uses counterfactuals to suppress

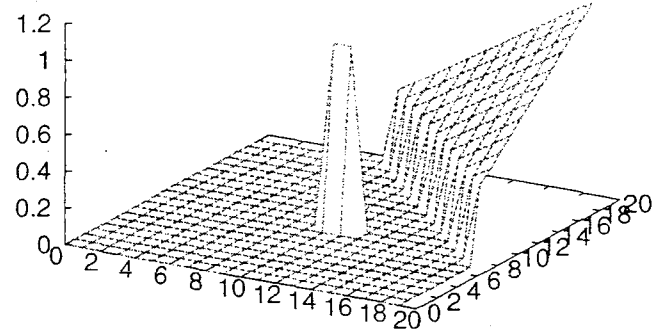


Figure 1: Distribution of Token Values in "Corner" World

the effect of other agents' actions:

$$\Delta Q_{UECR}(s_t, a_t) = \alpha(r_t(s_t) - r_t(s_t - s_{t,i} + s_b) + \max_a Q_{UECR}(s_{t+1}, a)),$$

where $s_t - s_{t,i} + s_b$ is the state resulting from removing agent i 's state and replacing it with the absorbing state s_b .

3.2 Results

To evaluate the effectiveness of QUICR-learning in the multi-agent Grid World, we conducted experiments on two different types of token distributions. The first set of tokens is designed to force congestion and tests the ability of QUICR-learning in domains where the reward function is far from being linearly separable. The second set is randomly generated from Gaussian kernels, to illustrate that the QUICR-learning capabilities in a non-hand crafted domain with spread out tokens (a domain favoring less dependent, local learners).

In all the experiments the learning rate was set to 0.5, the actions were chosen using an ϵ -greedy ($\epsilon = 0.15$) exploration scheme and tables were initially set to zero with ties broken randomly.

3.3 Corner World Token Value Distribution

The first experimental domain we investigated consisted of a world where the "highly valued" tokens are concentrated in one corner, with a second concentration near the center where the rovers are initially located. Figure 1 conceptualizes this distribution for a 20x20 world.

Figure 2a shows the performance for 40 agents on a 400 unit-square world for the token value distribution shown in Figure 1, and where an episode consists of 20 time steps (error bars of \pm one σ are included, though in most cases they are smaller than the symbols). The performance measure in these figures is sum of full rewards ($r_t(s_t)$) received in an episode, normalized so that the maximum reward achievable is 1.0. Note all learning methods are evaluated on the same reward function, independent of the reward function that they are internally using to assign credit to the agents.

The results show that local Q-learning generally produced poor results. This problem is caused by all agents aiming to acquire the most valuable tokens, and congregating towards the corner of the world where such tokens are located. In essence, in this case agents using local Q-learning competed, rather than cooperated. The agents using standard Q-learning did not fare better, as the agents were plagued by

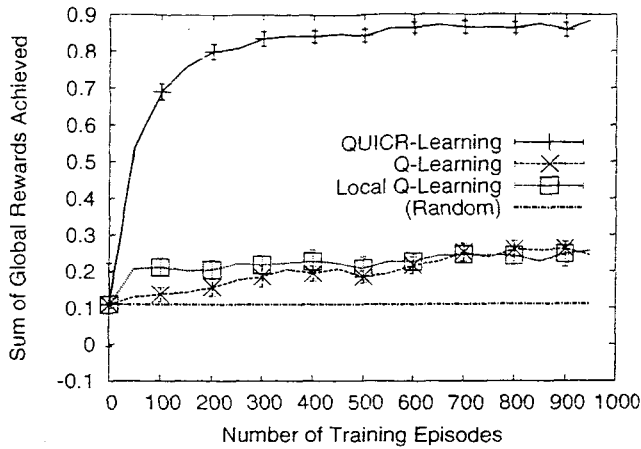


Figure 2: Learning Rates in Corner World with 40 Agents

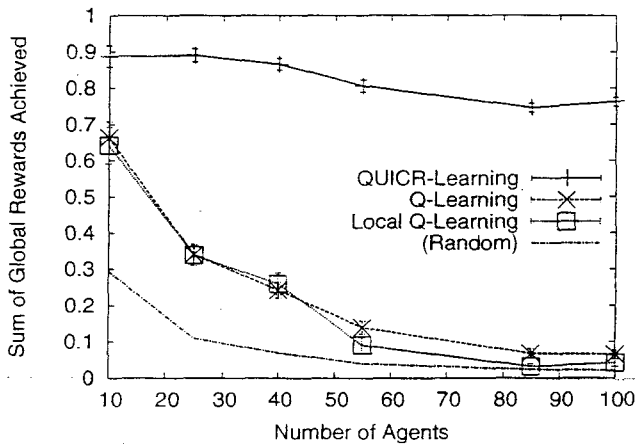


Figure 3: Scaling Properties of Different Payoff Functions.

the credit assignment problem associated with each agent receiving the full world reward for each individual action they took. Notice though that though local Q learning agents hit a performance plateau early, the standard Q-learning agents continue to learn, albeit slowly. This is because standard Q-learning agents are aligned with the system reward, but have low agent sensitivity. Agents using QUICR-learning on the other hand learned rapidly, outperforming both local and standard Q-learning by a factor of six (over random rovers).

Figure 3 explores the scaling properties for each algorithm. As the number of agents was increased, the difficulty of the problem was kept constant by increasing the size of the gridworld, and allocating more time for an episode. Specifically the ratio of the number of agents to total number of grid squares and the ratio of the number of agents to total value of tokens was held constant. In addition the ratio of the furthest grid square from the agents' starting point to the total amount of time in an episode was also held constant (e.g., 40 agents, 20x20 grid, 20 steps, 100 agents, 32x32 grid, 32 time steps). The scaling results show that agents using both local and standard Q-learning deteriorate rapidly as the number of agents increases. Agents using QUICR-learning on the

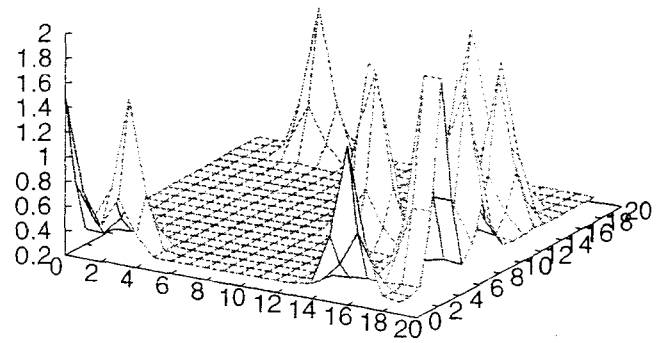


Figure 4: Distribution of Token Values in "Random" World

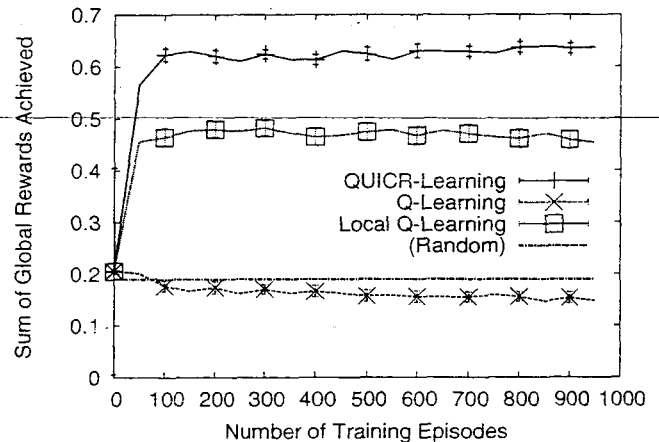


Figure 5: Learning Rates in Random World with 40 Agents

other hand were not strongly affected by the increase in the size of the problem, and outperformed local and standard Q-learners by a factor of thirty for the largest system. This is because QUICR-learning agents received rewards that were both aligned with the system goal and had high agent sensitivity (i.e., less affected by the size of the system). This result underscores the need for using rewards that suppress the affect of other agents actions in large systems.

3.4 Random World Token Value Distribution

In the second set of experiments, we investigate the behavior of agents in a gridworld where the token values are randomly distributed. In this world, for n agents, there are $n/3$ Gaussian 'attractors' whose centers are randomly distributed. Figure 4 shows an instance of the gridworld using this distribution for the 20x20 world, used in the experiments with 40 agents.

The results in Figures 5 and 6 show that agents using QUICR-learning are insensitive to changes in the token value distribution. Agents using local Q-learning perform significantly better in this case, showing a much larger sensitivity to the token distribution. The improvements are due to the spreading of tokens over a larger area, which allows agents aiming (and failing) to collect high valued tokens to still collect mid to low-valued tokens, surrounding the high valued tokens. In this domain, agents using standard Q-learning per-

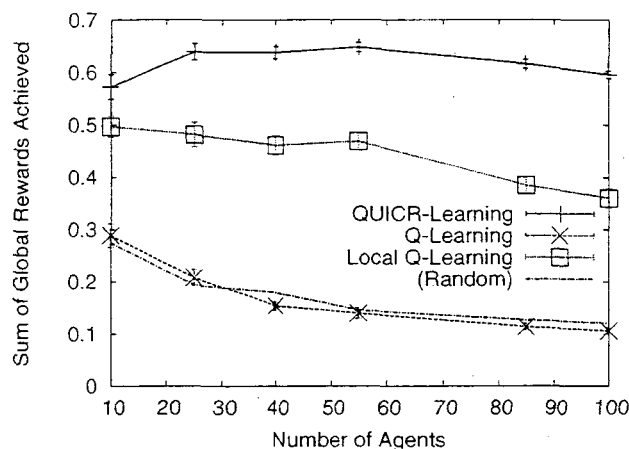


Figure 6: Scaling Properties of Different Payoff Functions.

formed particularly poorly. Indeed they were outperformed by agents performing random walks for systems of 40 agents. Due to the distributed tokens and the large number of agents, agents using standard Q-learning were never able to form an effective policy. These agents essentially moved from their initial random walk to a slightly more coordinated random walk, causing them to spread out less than agents performing independent random walks, and thus perform slightly worse than random agents.

The scaling results show again that QUICR-learning performs well as the number of agents increases. The interesting result here is that QUICR-learning does better with 40 to 55 agents than with 10 agents (this is not a statistical quirk, but a repeated phenomenon in many different random token configurations). One potential cause is that the larger number of agents more efficiently explore the space without being beset by the problems encountered by the other learning algorithms. The scaling results confirm that standard Q-learners perform slightly coordinated random walks in this setting, performing ever so slightly worse than random in all cases with more than 25 agents. With this token distribution the rewards used in standard Q-learning appear to be no better than random rewards, even with ten agents. While local Q-learning performs better on this token distribution than the previous one, it still scales less gracefully than does QUICR-learning.

4 Discussion

Using Q-learning to learn a control policy for a single agent in a problem with many agents is difficult, because an agent will often have little influence over the reward it is trying to maximize. In our example problems, an agent's reward received after an action could be influenced by as many as 3200 other actions from other time-steps and other agents. Even temporal difference methods that perform very well in single agent systems will be overwhelmed by the number of actions influencing a reward in the multi-agent setting. To address this problem, this paper introduced QUICR-learning, which aims at reducing the impact of other agent's actions without assuming linearly separable reward functions. Within the Q-

learning framework QUICR-learning uses the difference reward computed with immediate counterfactuals. While eliminating much of the influence of other agents, this reward was shown mathematically to be aligned with the global reward: agents maximizing the difference reward will also be maximizing the global reward. Experimental results in two Grid World problems, confirm the analysis showing that QUICR-learning learns in less time than standard Q-learning, and achieves better results than Q-learning variants that use local rewards and assume linear separability. While this method was used with TD(0) Q-learning updates, it also naturally extends to TD(λ), Sarsa-learning and Monte Carlo estimation. In domains with difficult temporal credit assignment issues, the use of these other variants could be beneficial.

References

- [Agogino and Tumer, 2004] A. Agogino and K. Tumer. Unifying temporal and structural credit assignment problems. In *Proc. of AAMAS-04*, New York, NY, July 2004.
- [Chalkiadakis and Boutilier, 2003] G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: A bayesian approach. In *Proc. of AAMAS-03*, Melbourne, Australia, July 2003.
- [Guestrin et al., 2002] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proc. of 19th ICML*, 2002.
- [Hu and Wellman, 1998] J. Hu and M. P. Wellman. Multi-agent reinforcement learning: Theoretical framework and an algorithm. In *Proc. of the Fifteenth International Conference on Machine Learning*, pages 242–250, June 1998.
- [Jones and Mataric, 2003] C. Jones and M. J. Mataric. Adaptive division of labor in large-scale multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-03)*, Las Vegas, NV, July 2003.
- [Kaelbling et al., 1996] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
- [Nair et al., 2003] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proc. of the Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003.
- [Stone et al., 2005] P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 2005.
- [Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
- [Vickrey, 1961] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 1961.
- [Watkins and Dayan, 1992] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3/4):279–292, 1992.
- [Wolpert and Tumer, 2001] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.